

### **Crosstalk between VMs**

Alexander Komarov, Application Engineer Software and Services Group Developer Relations Division EMEA



2 September 2015

### Legal Disclaimer & Optimization Notice

- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
- Copyright© 2014, Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.
- The cost reduction scenarios described in this document are intended to enable you to get a better understanding of how the purchase of a given Intel product, combined with a number of situation-specific variables, might affect your future cost and savings. Nothing in this document should be interpreted as either a promise of or contract for a given level of costs.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804





### Contents

- Defining crosstalk
- Setup, what and how we measure
- Causes:
  - System
  - Cache
    - Splitlock
    - Capacity and conflict misses
    - CAT
  - Memory
  - PCIe
- Conclusion
- References





## What is crosstalk?

- Ways to heavily (~2x or more) impact performance of other VMs on the same host, w/o sharing execution cores and IO devices.
- Use cases :
  - Industrial workload consolidation
  - NFV
  - Cloud hosting
- Not malicious (DoS, Side channel attacks).



Consideration – resources shared by VMs





### Setup, what and how we measure

Xeon E5 3<sup>rd</sup> generation,

RHEL 7.1(3.10.0-229.el7.x86\_64), qemu/kvm 1.5.3

No HT, no power management (easier to measure)

VMs on dedicated cores.

TSC – native, virtualized

Measurements – TSC and PMU readings in micro benchmarks, linux perf, Intel Vtune.







### **Common Last Level Cache**

- Wbinvd ~ 15% (<2x)
- Split lock and uncachable lock
- Capacity misses/priority inversion
- Cache Allocation Technology
  - Conflict misses





## What is split lock, how to detect

// Get a split lock pointer

for( mylock = array; ((unsigned int)pointer % 64) != 62; pointer++);

// infinite disturber loop

while (1)  $\{$ 

// InterlockedIncrement(mylock); // We are not in Windows
asm ("lock;incl (%0) " : "=r" (mylock));

// Make it twice and perf hit doubles.

How to detect: LOCK\_CYCLES.SPLIT\_LOCK\_UC\_LOCK\_DURATION event on offender core, higher MACHINE\_CLEARS.MEMORY\_ORDERING – on other cores.

Uncachable lock too...





### **Crosstalk from a splitlock**



X – linked list size(log scale) Y – TSC cycles per element. Y(x) = P(xL1)\*L1I+P(xL2)L2I+P(xL3)\*L3I+P(xRAM)\*RAMUncore just stopsl





### **Capacity misses**

Classical benchmark: STREAM on one core, whatever cyclical workload on another.



Events: L1D load misses, L2 misses, LLC misses





## Intel<sup>®</sup> Xeon<sup>®</sup> Processor E5-2600v3 Feature Overview – CMT and CAT

#### Cache Monitoring Technology (CMT)

- Identify misbehaving or cache-starved applications and reschedule according to priority
- Cache Occupancy reported on per Resource Monitoring ID (RMID) basis



#### **Last Level Cache**

#### Cache Allocation Technology (CAT)

- Available on Communications SKUs only
- Last Level Cache partitioning mechanism enabling the separation of applications, threads, VMs, etc.
- Misbehaving threads can be isolated to increase determinism



CAT presentation is at 14:00 on Tuesday





### CAT and LLC coloring







### **CAT and conflict cache misses**



Iterating over linked list [size == LLC part permitted]

LLC misses% = mandatory + capacity + conflict(ways) Mandatory == 0, capacity = const %, coflict ~= f(1/ways)





# **Sharing memory bandwidth**



Events: LLC miss.Local\_dram (and make sure .remote\_dram is 0)





### **PCIe**

Root complex is shared across cores in a socket. Possible to slow down other VM's IO requests but can't measure w/o precise expensive tools. Not relevant for most applications. SR-IOV is very useful but old news. PCIe Virtual channels (vc0, vc1) are supported in Xeon E5 v2+ but if you don't count hundreds of nanosecond jitter – don't bother!





## Measured using Intel<sup>®</sup> VTune<sup>™</sup> Amplifier for Systems

- Acts a s linux perf frontend and a visualizer
- New support for KVM guest os performance tuning

🛿 Advanced Hotspots 1 Hotspots viewpoint (change) ③ Intel VTune Amplifier 2016 for Systems														stems				
< 📰 Collection Log 😝 Analysis Target 🛱 Analysis Type 🕅 Summary 🚱 Bottom-up 🗳 Caller/Callee 🕏 Top-down Tree 🛃 Platform 🗟 Grap														Grapl 🕨				
Gr	ouping:	Functio	tion / Thread / H/W Context / Call Stack								\$	<b>1</b> 40	Q	*		Data Of Interest	: (CPU Metrics	;) 🗸
R,	Function /	ion / Thread /		CPU Time•				*	Instructions Retired	CPI Rate	CPU Fre		Modul		_	★Viewing 4 1 of 1 ▷ selected stack(s 100.0% (11.010s of 11.010s)		
H/W Context / Ca		kt / Call S	t	Effective Time			Spin	0v							Ξ			
▶[Outside any known r			n n	11.010s					47,411,000,000		0.923					[Unknown]![Out	sile] - [Unkı	nown]
Þsystem_call				1.614s			0.108s	0s	3,115,000,000	1.608	0.833	vmli	inux			[Unknown]![Unk	(nown] - [Unk	nown]
∮func@0x228ef0				0.270s			0s	0s	2,408,000,000	0.292	0.747	qemu-system				[Unknown]![Unknown] - [Unknown]		
▷choose_wl_class_and			nd_	0.201s			0s	0s	28,000,000	21.125	0.841	vmli	inux [	gues				
Þipv6_find_hdr			0.127s			0s	0s	297,500,000	1.000	0.669	vmli	inux [	gues					
▶ipv6_skip_exthdr				0.0985			0s	0s	189,000,000	1.537	0.847	vmli	inux [	gues				
▶atomic_cmpxchg				0.077s			0s	0s	10,500,000	28.333	1.104	vmli	inux [	gues				
¢vmx_vcpu_run			0.076s			0s	0s	3,500,000	69.000	0.908	kvm	inte	l.ko					
▶hpet_cpuhp_noti		np_notify		0.061s			0s	0s	241,500,000	0.884	1.000	vmli	inux [	gues				
D migrate task Selected 1 ro		ed 1 row	(s):	0.055c		11.010s	05	0s	133 000 000	1 368	0.923	vmli	inuv I	71100				
<			><				05	00	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	0.7.10	0.020			>	ت			
	Q <b>∞Q+</b> Q−Q#			5s 10		ls		15s	20s		24.958		Bs	1	30s	Thread		
	kvm (TID	D: 6	N V		1 100	1. 20 Miles	-		Aurilianurun utrarte la	Here the also		JU UD	n i	NV.				
	kvm (TIC	D: 6	<b></b> )i	17-00-0		Million i		- 		And and a second		1		3064	¢۲		Rur	nning
ad	kvm (TID	D: 6		4644	n is m	Cipation In-							~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	a o 🗢	đ i			U Ti
hrei	kvm (TIC	D: 6 🔽	÷γ							مربع المربع المربع			D 000≍	30 O			🗹 🚧 Spi	n an 🗉
F	kvm (TID: 6					~ ~ ~					2 🗸	. △	- in	,		······	□ ♥ Hard	war
	kvm (TID: 6																🗹 🗖 Tas	sk
	kum (TIP									1						· ·	🗌 🕇 Tra	nsiti
	CPU Time	e												CPU Tim	ne 🗖			
		<	Ш													) »	CPI	U Ti 🗸
🛪 🛛 No filters are applied. 💝 Call Stack Mode: User functions + 1 🔍 Inline Mode: Terminal pop Mode: Functions only 🔽																		





## **Conclusion (sorted)**

- Run VMs on different cores or better sockets, mind the HT
- Use CAT when available. Mind the conflict misses. Cache coloring when absolutely necessary.
- Use COD on Xeon.
- Check for split locks/uncachable locks.





### **References:**

- Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3, 17.5 for CAT.
- Linuxcon Europe 2015: Monitoring and Controlling Cache Allocation For Quality of Service Guarantees in Linux - Matt Fleming, Intel
- Linuxcon Europe 2011: Using Cache Coloring for LLC Partitioning for More Predictable Performance on x86 Platforms, Alexander Komarov, Intel
- Xeon E5 v2 1600-2600 public Datasheet



